Exercise #3

Fortgeschrittene Statistische Software für NF - SS 2022/23

Bichmai Vu (12057445), Nele Steiner (12548686)

2023-06-08

Exercise 1: Initializing git (4 Points)

For this whole exercise sheet we will be tracking all our changes to it in git.

a) Start by initializing a new R project with git support, called `exeRcise-sheet-3`. If you forgot how to do this, you can follow this guide.

b) Commit the files generated by Rstudio.

c) For all of the following tasks in this exercise sheet we ask you to always commit your changes after finishing each subtask e.g. create a commit after task *1d*, *1e* etc.

d) Name 2 strengths and 2 weaknesses of git.

strengths: 1. It's very hard to lose files with git. 2. Git allows easy collaboration as it allows everyone who is working on a project to see the changes other persons did.

weaknesses: 1. Git struggles with large files and a bit. 2. Git needs to be used explicitly. It doesn‘t just work in the background.

e) Knit this exercise sheet. Some new files will automatically be generated when knitting the sheet e.g. the HTML page. Ignore these files, as we only want to track the source files themselves.

Exercise 2: Putting your Repository on GitHub (3.5 Points)

For this task you will upload your solution to GitHub.

a) Create a new repository on GitHub in your account named `exeRcise-sheet-3`. Make sure you create a **public repository** so we are able to see it for grading. Add the link to the repository below: https://github.com/bichmaivu/exeRcise-sheet-3.git
b) Push your code to this new repository by copying and executing the snippet on github listed under `...or push an existing repository from the command line`.
c) Regularly push your latest changes to GitHub again and especially do so when you are finished with this sheet.

Exercise 3: Baby-Names in Munich (4.5 Points)

Download the latest open datasets on given names ("Vornamen") from the open data repository of the city of Munich for the years 2022 and 2021.

Link: https://opendata.muenchen.de/dataset/vornamen-von-neugeborenen

a) Download the data for both years and track it in git. For small datasets like these adding them to git is not a problem.

b) Load the data for both years into R. Check the type of the count variable ("Anzahl") and look into the data to determine why it is not numeric? Fix the problem in an appropriate manner, it is OK if some of the counts are inaccurate because of this. Explain your solution and the repercussions.

```
vornamen_2022 <- read.csv("data/open_data_portal_2022.csv")
vornamen_2021 <- read.csv("data/vornamen_2021.csv")

vornamen_2021 <- vornamen_2021 %>%
  rename(first_name = Vorname,
         number = Anzahl,
         gender = Geschlecht
    )

vornamen_2022 <- vornamen_2022 %>%
  rename(first_name = Vorname,
         number = Anzahl,
         gender = Geschlecht
    )

typeof(vornamen_2021$number)
```

```
## [1] "character"
```

```
typeof(vornamen_2022$number)
```

```
## [1] "character"
```

```
vornamen_2021[vornamen_2021=="4 oder weniger"] <- "4"
vornamen_2022[vornamen_2022=="4 oder weniger"] <- "4"

vornamen_2021$number = as.numeric(vornamen_2021$number)
is.numeric(vornamen_2021$number)
```

```
## [1] TRUE
```

```
vornamen_2022$number = as.numeric(vornamen_2022$number)
is.numeric(vornamen_2022$number)
```

```
## [1] TRUE
```

The variable type of "number" is "character" in both years. This is the case, because for some names it isn't possible to say for sure how often the name has been used, so there is no exact number in the datasets. The value "4 or less" explains why the variable type "numeric" wasn't used. We rounded the affected values to 4 up. However this has the effect that it is assumed now that there were 4 namings, although we cannot say this with certainty.

Note: We renamed the columns, to use a consistent language in the datasets.

c) Calculate the total number of babies born in Munich in 2022 and 2021. Which year had the bigger baby-boom?

```
total_number_of_babies_2021 <- sum(vornamen_2021$number)

total_number_of_babies_2022 <- sum(vornamen_2022$number)

total_number_of_babies_2021
```

```
## [1] 26620
```

```
total_number_of_babies_2022
```

```
## [1] 25199
```

The bigger baby-boom has the year 2021 with 26620 babies.

d) Add a new column `year` to both datasets which holds the correct year for each.

```
vornamen_2021 <- vornamen_2021 %>%
        mutate(year = 2021)
vornamen_2022 <- vornamen_2022 %>%
        mutate(year = 2022)
```

e) Combine both datasets into one using `bind_rows()`.

```
combination_2021_2022 <- bind_rows(vornamen_2021, vornamen_2022)
```

f) Combine the counts for same names to determine the most popular names across both years. Print out the top 10 names in a nicely formatted table for both years. Include a table caption.

```
most_popular_names <- combination_2021_2022 %>%
  group_by(first_name) %>%
  summarise(total_number = sum(number)) %>%
  arrange(desc(total_number)) %>%
  head(10)

knitr::kable (most_popular_names,
              caption = "The 10 most used names in 2021 and 2022 combined")
```

Table 1: The 10 most used names in 2021 and 2022 combined

first_name	total_number
Maximilian	240
Emilia	234
Felix	220
Anton	206
Emma	199
Leon	195
Noah	185
Jakob	180
Anna	178
Lukas	173

Exercise 4: Chat GPT + apply (3 points)

For this task: Specifically use ChatGPT to solve the task and submit your prompts in addition to the solution

a) The code below does not work because the wrong apply function has been used. Find out which apply function would be correct and why it did not work. Correct the code. Also calculate the rowwise means.

###Create a sample data frame

tax_data <- data.frame(Name = c("Munich GmbH", "ABC Inc.", "Backpacks 1980", "Bavarian Circus"), Tax_2019 = c(5000, 4000, 6000, 3500), Tax_2020 = c(4800, 4200, 5800, 3700), Tax_2021 = c(5200, 3800, 5900, 3400))

###Calculate column-wise means

column_means <- lapply(tax_data

$$, -1$$

, 2, mean)

column_means

```
# Corrected code by chat gbt
# Create a sample data frame

tax_data <- data.frame(
  Name = c("Munich GmbH", "ABC Inc.", "Backpacks 1980", "Bavarian Circus"),
  Tax_2019 = c(5000, 4000, 6000, 3500),
  Tax_2020 = c(4800, 4200, 5800, 3700),
  Tax_2021 = c(5200, 3800, 5900, 3400)
)

column_means <- colMeans(tax_data[, -1])
column_means
```

```
## Tax_2019 Tax_2020 Tax_2021
##     4625     4625     4575
```

There were two issues in the original code: 1. The assignment operator "<-" was incorrectly written as "<-". The correct operator is "<-", which is used for assigning values to variables in R. 2. The "lapply" function call had incorrect arguments. The code wasn't correct, because of presentation "

$$, -1$$

" and we need to correct this into "[, -1]". Also instead of 2, as the second argument need to pass the mean function. So the correct call should be "lapply(tax_data[, -1], mean)".

b) Using ChatGPT try to understand what the rapply() function does. Create an easy example with mock data where the function is used and explain it in your words.

Explanation: The "rapply" function applies a function to each element of a list or nested list structure, and the "rapply" function traverses the structure recursively of "lapply", applying the function to all levels of nesting.

```
my_list <- list(a = 1:3, b = 4:6, c = 7:9)

rapply(my_list, function(x) x^2)
```

```
## a1 a2 a3 b1 b2 b3 c1 c2 c3
##  1  4  9 16 25 36 49 64 81
```

Explanation: In this example "rapply" is used to take each value of "my_list" to square.

Final Note

Make sure to push all your commits and changes to GitHub before submitting the exercise sheet.

Note: We both worked on the exercise sheet independently and each have our own repository. The changes now are only the final ones.